Can We Remove the Square-Root in Adaptive Gradient Methods? A Second-Order Perspective

Wu Lin, Felix Dangel, Runa Eschenhagen, Juhan Bae, Richard Turner, Alireza Makhzani

University of Toronto · Vector Institute · University of Cambridge

Paper · Code

Many adaptive methods can be interpreted as diagonal second-order methods with an extra square root. We observe that removing the root, thus strengthening the second-order perspective, does not harm their performance when proper changes are made.

Removing the root also allows us to design matrix methods that don't require inversions, e.g. a faster version of Shampoo.

Which Square Root Are We Talking About?

Many adaptive optimizers use statistics $\mathbf{S}$ in the form of the gradient outer product (GOP) $\mathbf{g}\mathbf{g}^\top$. The inverse square root of these statistics is then used to update the weights.

AdaGrad:

$$\mathbf{S} \leftarrow \mathbf{S} + \beta_2 \mathbf{g}\mathbf{g}^\top$$
$$\mu \leftarrow \mu - \beta_1 \mathrm{diag}(\mathbf{S})^{-1/2}\mathbf{g}$$

RmsProp:

$$\mathbf{S} \leftarrow (1-\beta_2)\mathbf{S} + \beta_2 \mathbf{g}\mathbf{g}^\top$$
$$\mu \leftarrow \mu - \beta_1 \mathrm{diag}(\mathbf{S})^{-1/2}\mathbf{g}$$

RF-AdaGrad (ours):

$$\mathbf{S} \leftarrow \mathbf{S} + \beta_2 \alpha \mathbf{g}\mathbf{g}^\top$$
$$\mu \leftarrow \mu - \beta_1 \mathrm{diag}(\mathbf{S})^{-1}\mathbf{g}$$

RF-RmsProp:

$$\mathbf{S} \leftarrow (1-\beta_2)\mathbf{S} + \beta_2 \alpha \mathbf{g}\mathbf{g}^\top$$
$$\mu \leftarrow \mu - \beta_1 \mathrm{diag}(\mathbf{S})^{-1}\mathbf{g}$$

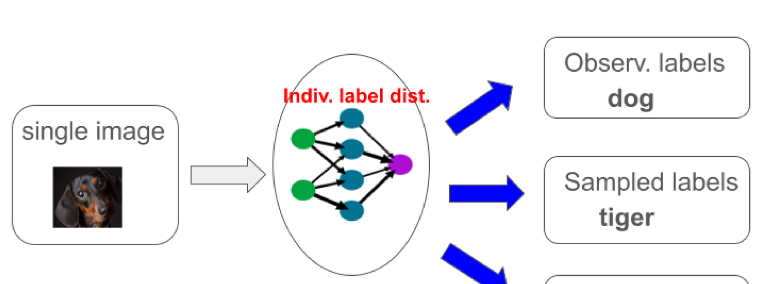
Why remove it?

- Adaptive methods under-perform SGD on CNNs. Could it be due to the root?
- Make them more similar to natural-gradient methods that use the empirical Fisher (EF)

Our contributions

1. **[Empirical]** In modern training setups, root-free methods perform as well as SGD on CNNs
2. **[Theoretical]** Provide a clean interpretation of $\mathbf{g}\mathbf{g}^\top$ as EF and preserve invariance
3. **[Practical]** Removing roots and inversions in methods with non-diagonal $\mathbf{S}$

Clean Interpretation: Empirical Fisher as (Scaled) GOP



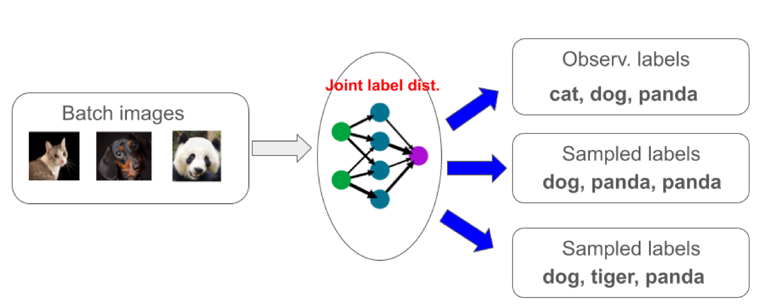
Standard empirical Fisher

Distribution factorized over **individual** labels

$$\sum_{i=1}^{N} \mathbb{E}_{y_i \sim p(y_i|\mathbf{x}_i)} \left[\nabla_\mu \log p(y_i \mid \mathbf{x}_i) \nabla_\mu^\top \log p(y_i \mid \mathbf{x}_i)\right]$$

expectation approximation

$$\sum_{i=1}^{N} \left[\nabla_\mu \log p(y_i \mid \mathbf{x}_i) \nabla_\mu^\top \log p(y_i \mid \mathbf{x}_i)\right] = \sum_{i=1}^{N} \mathbf{g}_i \mathbf{g}_i^\top$$



Our empirical Fisher

Distribution over **joint** labels

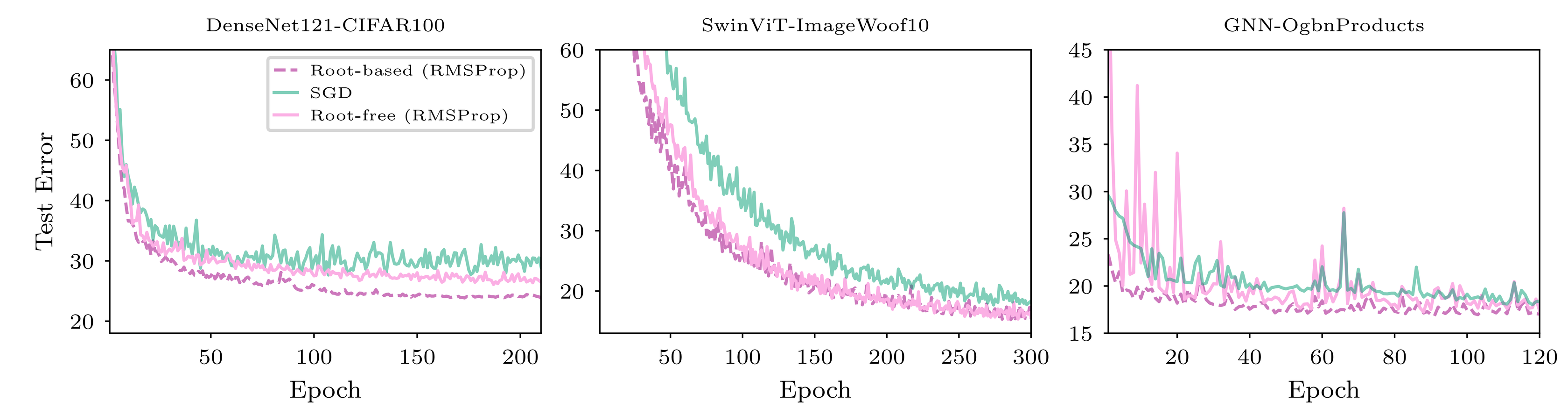
$$\mathbb{E}_{\mathbf{y} \sim p(\mathbf{y}|\mathbf{X})} \left[\nabla_\mu \log p(\mathbf{y} \mid \mathbf{X}) \nabla_\mu^\top \log p(\mathbf{y} \mid \mathbf{X})\right]$$

expectation approximation

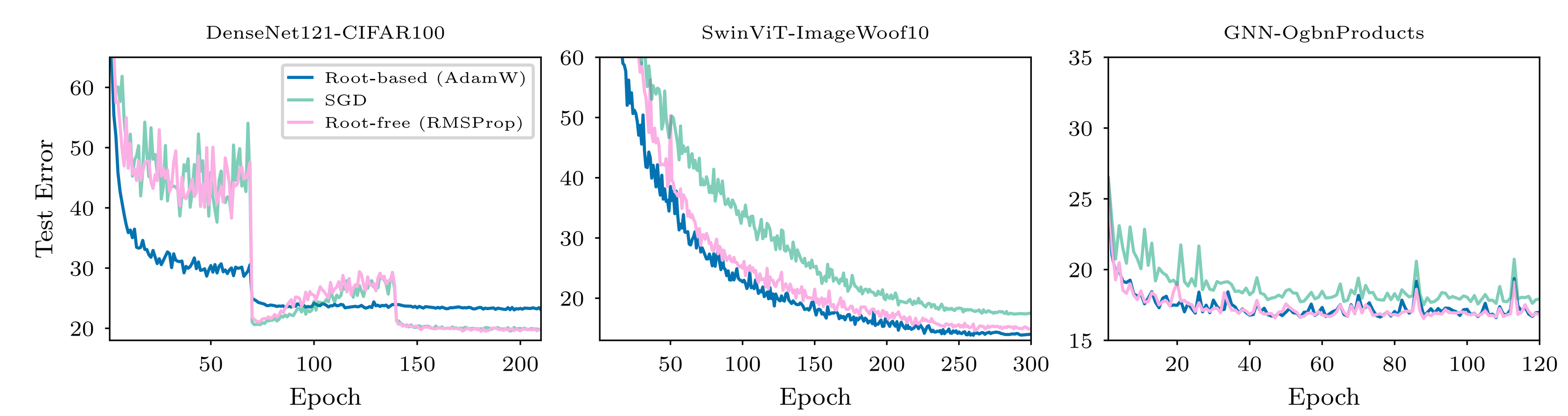
$$\nabla_\mu \log p(\mathbf{y} \mid \mathbf{X}) \nabla_\mu^\top \log p(\mathbf{y} \mid \mathbf{X}) = \mathbf{g}\mathbf{g}^\top$$

Empirical Observations When Removing the Square Root

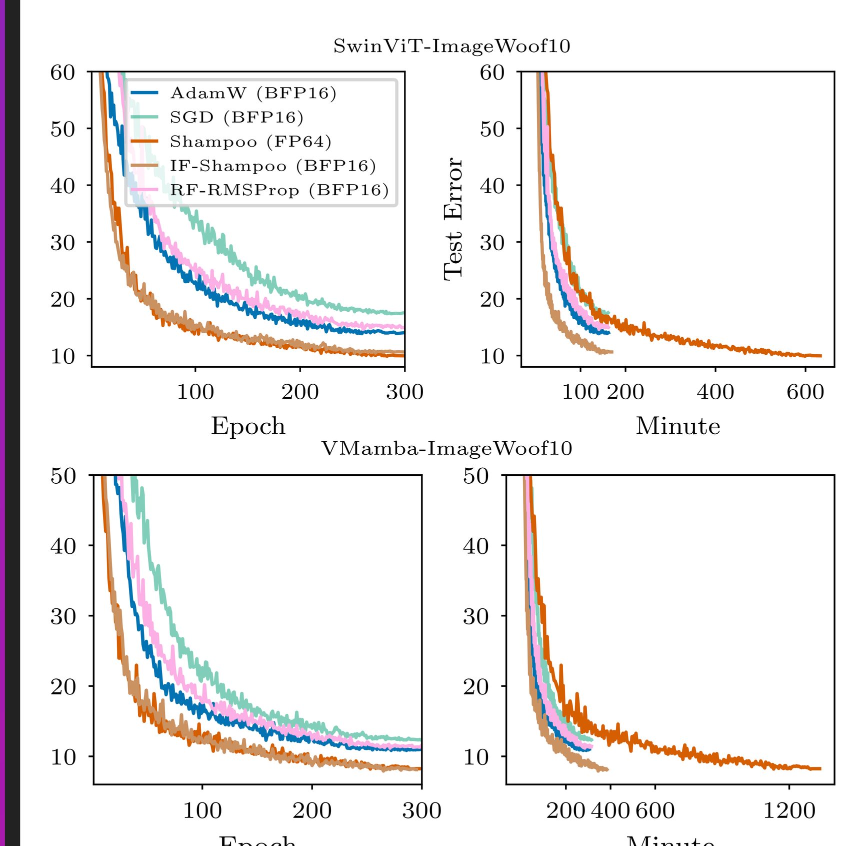
Simply removing the root does not improve the performance of root-free methods.



Through a second-order perspective, we identify some fixes to make them work well.



1. **[Standard nowadays]** Non-constant learning rate schedule
2. **[Bottom left box]** Additional scaling, because densities for the Fisher must be normalized (α = 1 for sum; α = B for average)
3. **[Bottom right box]** Non-zero init. of preconditioner $\mathbf{S}$, because it can be viewed as inverse covariance of a Gaussian



Inverse- and root-free updates accelerate Kronecker-based methods based on $\mathbf{g}\mathbf{g}^\top$ in half precision.

Want to try the optimizer?

```
pip install sirfshampoo
```

Full-matrix and Kronecker-structured Adaptive Methods

Full-matrix methods aim to use $\mathbf{S}$ rather than diag($\mathbf{S}$), e.g. full-matrix AdaGrad:

$$\mathbf{S} \leftarrow \mathbf{S} + \beta_2 \mathbf{g}\mathbf{g}^\top \qquad \mu \leftarrow \mu - \beta_1 \mathbf{S}^{-1/2}\mathbf{g}$$

But $\mathbf{S}$ is usually infeasible to store, which is why most methods impose additional structure in their pre-conditioner, e.g. Kronecker structure in Shampoo ($\mathbf{G}$:= Mat($\mathbf{g}$)):

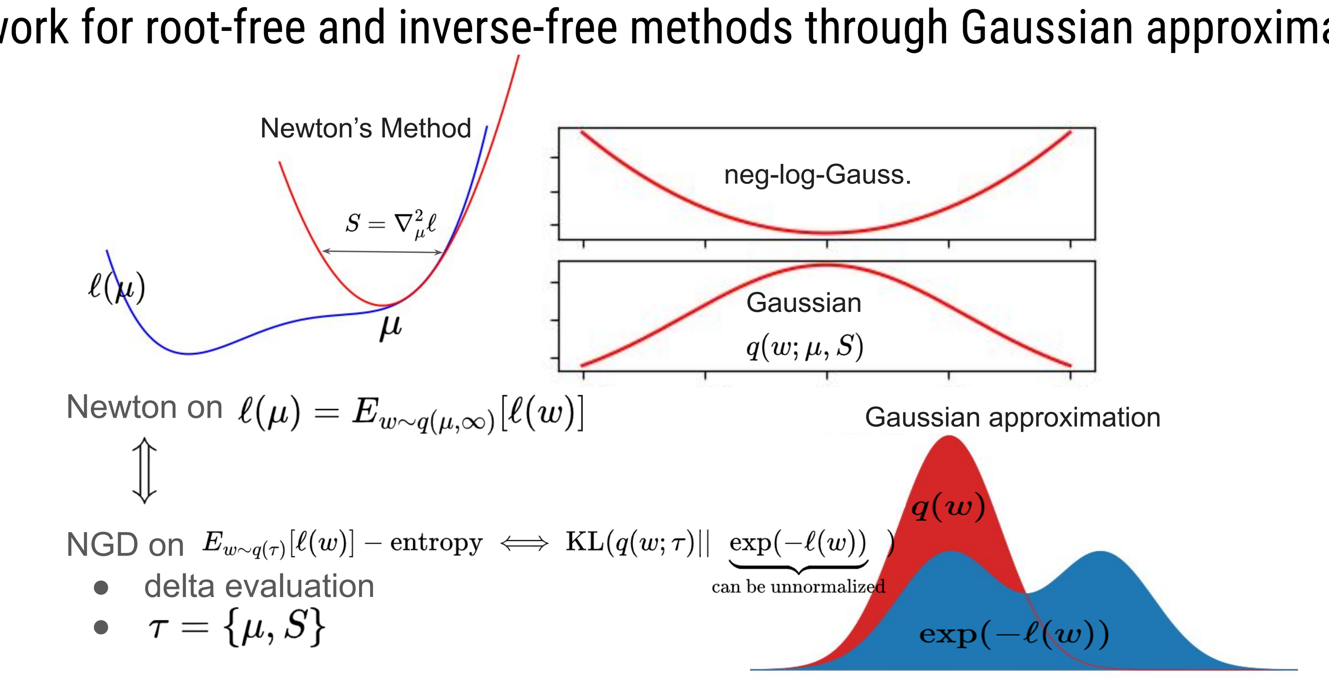
$$\mathbf{S}_1 \leftarrow (1-\beta_2)\mathbf{S}_1 + \mathbf{G}\mathbf{G}^\top \qquad \mu \leftarrow \mu - \beta_1 (\mathbf{S}_1^{1/2} \otimes \mathbf{S}_2^{1/2})^{-1/2}\mathbf{g}$$
$$\mathbf{S}_2 \leftarrow (1-\beta_2)\mathbf{S}_2 + \mathbf{G}^\top\mathbf{G}$$

As the pre-conditioner now differs from $\mathbf{g}\mathbf{g}^\top$, we need to decouple the concepts of **pre-conditioner** and **curvature approximation**.

We provide a unified recipe to incorporate any curvature info (e.g., $\mathbf{g}\mathbf{g}^\top$) into pre-conditioners with structure, and show how to make the update inverse-free.

Removing Inversions Through Variational Gaussian Approximation

A framework for root-free and inverse-free methods through Gaussian approximations



Our framework:

- Preconditioner $\mathbf{S}$: inverse covariance
- Curvature $\mathcal{H} = \nabla_\mu^2 \ell$: partial derivative w.r.t. covariance

Inverse-free methods via preconditioner invariance:

- Inverse-free: NGD on $\{\mu, \mathbf{S}^{-1}\}$ or $\{\mu, \mathbf{B}\}$, where $\mathbf{S}^{-1} = \mathbf{B}\mathbf{B}^T$
- Arbitrary curvature : e.g., $\mathcal{H} = \alpha \mathbf{g}\mathbf{g}^\top$; (sum: α = 1), (average: $\alpha = B$)
- Generic structural projection via chain rule: $\mathcal{H}$ as a partial derivative
- Kronecker structured: Root-free Shampoo as NGD on $\{\mu, \mathbf{B}_1, \mathbf{B}_2\}$, where $\mathbf{S}^{-1} = (\mathbf{B}_1\mathbf{B}_1^\top) \otimes (\mathbf{B}_2\mathbf{B}_2^\top)$